UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/628,960	07/28/2003	Carlos Bonilla	200309109-1	6175

22879 7590 12/28/2007

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
DAO, THUY CHAN	
ART UNIT	**PAPER NUMBER**
2192	

NOTIFICATION DATE	DELIVERY MODE
12/28/2007	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

JERRY.SHORMA@HP.COM
mkraft@hp.com
ipa.mail@hp.com

mN

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/628,960
Filing Date: July 28, 2003
Appellant(s): BONILLA, CARLOS

John P. Wagner, Jr.
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed October 1, 2007 appealing from the Office action mailed April 30, 2007.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

(9) Grounds of Rejection

The following grounds of rejection are applicable to the appealed claims:

☐ Claims 1-20 are rejected under 35 U.S.C. 102(e) as being anticipated by Evans (art of record, US Patent No. 6,826,746).

Claim 1:

Evans discloses *an emulation and native language interface test method comprising:*

initializing an emulation language virtual machine (e.g., FIG. 1, initializing JVM 16, which is running the ICAT2 probe 41, col.5: 7-12)*;*

wrapping native language code in a simulation test macro which creates simulated interfacing problems (e.g., col.1: 7-11, debugging a Java application that includes native method dynamic load libraries; FIG. 4, col.5: 65-66 and col.6: 3-4, "simulation test macro" as probe function/launch method 46; col.5: 64 - col.6: 5, wrapping/containing native language code C/C++; col.2: 15-22, a first "daemon" process that performs native method debugging; col.3: 66 - col.4: 7)*; and*

examining reaction to said simulated interfacing problems when an emulation language application is run (e.g., FIG. 6, Interface Code Analysis Tool 41 (ICAT2) with event handlers 61-67 (examining reaction to said simulated interfacing problems); GUI of ICAT, ICAT2 to set breakpoints, step applications, examine application stack and variables, col.2: 33-47; col.8: 15-36).

Claim 2:

The rejection of claim 1 is incorporated. Evans also discloses *said emulation language virtual machine creates a runtime environment and said runtime environment can include a class loader subsystem and an execution engine* (e.g., FIG. 1, class loader subsystem in JVM 16, engine 13, col.5: 2-17).

Claim 3:

The rejection of claim 1 is incorporated. Evans also discloses *said simulated test problems include simulations of error conditions associated with a native language code method attempt to respond to a call from emulation language code* (e.g., FIG. 6, ICAT2 with event handlers 61-67; GUI of ICAT, ICAT2 to set breakpoints, step applications, examine application stack and variables, col.2: 33-47).

Claim 4:

The rejection of claim 1 is incorporated. Evans also discloses *forwarding an indication that there is a insufficient memory allocation exception to a native language method attempting to ascertain an indication of a memory location for information associated with a native language function* (e.g., col.8: 1-8, Memory Usage Information).

Claim 5:

Evans discloses *a Java Native Language Interface testing system comprising:*

means for communicating information (e.g., FIG. 6, Interface Code Analysis Tool 41 (ICAT2) with event handlers 61-67; col.8: 15-36; GUI of ICAT, ICAT2 to set breakpoints, step applications, examine application stack and variables, col.2: 33-47; col.3: 66 – col.4: 60)*;*

means for processing said information, including instructions for testing a Java Native Language Interface, said means for processing said information coupled to said means for communicating information (e.g., col.1: 7-11; col.2: 15-22)*; and*

means for storing said information, including said instructions for testing said Java Native Language Interface, said means for storing said information coupled to said means for communicating information (e.g., col.6: 52-57; col.8: 15-19; col.8: 25-col.9: 67; FIG. 13, data sent to application, data output from application; col.4: 65 – col.5: 17; col.5: 18-46).

Claim 6:

The rejection of claim 5 is incorporated. Evans also discloses *said means for processing performs a Java Native Language Interface test method* (e.g., FIG. 1, ICAT2 Probe 41, col.5: 2-12).

Claim 7:

The rejection of claim 5 is incorporated. Evans also discloses *an interface testing macro module* (e.g., FIG. 1, JDaemon DLL 14, col. 5: 5-8; col.2: 15-22)

Claim 8:

The rejection of claim 5 is incorporated. Evans also discloses *emulating a Java virtual machine* (e.g., FIG. 15, JVM 16, col.4: 40-51).

Claim 9:

Evans discloses *a Java Native Language Interface test method comprising:*

investigating Java Native Language Interface test mode status (e.g., col.5: 62 – col.6: 5, if debugging is enabled, calling "launh_then_attach", passing arguments to each connector, ..., setting "suspend" argument to true)*;*

running a Java application with simulated Java Native Language Interface problems if said Java Native Language Interface test mode is enabled (e.g., FIG. 1, col.5: 7-12; col.1: 7-11; col.2: 15-22)*; and*

initiating a call to a Java Native Language Interface function directly without said simulated Java Native Language Interface problems if said Java Native Language Interface test mode is not enabled (e.g., col.1: 32-37, Java application with native method DLL's, which calls native method dynamic load libraries directly when executing; col.1: 66 – col.2: 5, said application executes with debugging enabled or not enabled).

Claim 10:

The rejection of claim 9 is incorporated. Evans also discloses *said Java Native Language Interface test mode status indicator indicates if said Java Native Language*

Interface test mode status is enabled (e.g., FIG. 1, JVM 16, col.5: 7-12; col.1: 7-11; col.2: 15-22).

Claim 11:

The rejection of claim 9 is incorporated. Evans also discloses *said Java Native Language Interface test mode status indicator is a flag wherein a state of said flag indicates if said Java Native Language Interface test mode status is set* (e.g., col.5: 62 – col.6: 5).

Claim 12:

The rejection of claim 9 is incorporated. Evans also discloses *a register value indicates said Java Native Language Interface test mode status* (e.g., col.5: 62 – col.6: 5).

Claim 13:

The rejection of claim 9 is incorporated. Evans also discloses *identifying indications of Java Native Language Interface code trouble associated with out of memory situations* (e.g., col.8: 1-8, Memory Usage Information).

Claim 14:

The rejection of claim 9 is incorporated. Evans also discloses *a Java Native Language Interface problem simulation process is performed to simulate Java Native Language Interface problems* (e.g., col.4: 40-51).

Claim 15:

The rejection of claim 9 is incorporated. Evans also discloses:

determining a Java Native Language Interface problem simulation occurrence level (e.g., col.7: 52-63);

introducing simulation randomness (e.g., col.6: 10-21);

performing an analysis whether to initiate a simulation of Java Native Language Interface problem; calling a Java Native Language Interface memory allocation function normally (e.g., col.8: 2-8)*;*

forwarding a Java Native Language Interface problem indicator automatically; and implementing a reaction to the Java Native Language Interface problem indication (e.g., col.8: 63 – col.9: 14).

Claim 16:

The rejection of intervening claim 15 is incorporated. Evans also discloses:

looking up a predefined Java Native Language Interface problem simulation occurrence level (e.g., col.8: 2-8)*;*

generating a random value (e.g., col.6: 10-21)*; and*

correlating said random value to said JNI problem simulation occurrence level (e.g., col.7: 52-63).

Claim 17:

The rejection of intervening claim 16 is incorporated. Evans also discloses:

comparing said randomly generated value to said Java Native Language Interface problem simulation occurrence level (e.g., col.6: 10-21)*; and*

initiating a simulation of a Java Native Language Interface problem if said generated value from is less than said Java Native Language Interface problem simulation occurrence level (e.g., col.8: 63 – col.9: 14).

Claim 18:

The rejection of intervening claim 16 is incorporated. Evans also discloses *initiating a controlled shut down* (e.g., col.5; 62 – col.6: 5).

Claim 19:

The rejection of intervening claim 16 is incorporated. Evans also discloses *clearing a system and canceling information inventory collections that is occupying memory space* (e.g., col.8: 2-8).

Claim 20:

The rejection of intervening claim 16 is incorporated. Evans also discloses *providing an indication of the Java Native Language Interface problem to a user* (e.g., col.8: 63 – col.9: 14).

(10) Response to Argument

1. Whether Claims 1-4 are Anticipated Under 35 USC 102(e) by Evans (Brief, pp. 8-12):

A) Cited Art is not "Arranged as in the Claim" (Brief, pp. 8-9):

The examiner respectfully disagrees with Appellant's assertions. As an initial matter, the examiner notes that these issues have been raised and fully responded in the previous Office action mailed April 30, 2007.

In the Remarks previously filed February 6, 2007, The Appellant asserted, "...the present rejection picks and chooses information from a variety of locations and embodiments/examples within the Evans reference in an attempt to assemble the elements of the invention as claimed ..." (Remarks, page 11: 24-26, page 12: 25 – page 13: 2, and page 13: 21-25).

In contrast with Appellant' assertions, Evans explicitly set forth:

> "...By modifying the currently-available ICAT to use JPDA, users of the new ICAT have access to the new features immediately and in future ICAT releases. For clarity and distinction, the existing debugger tool will be referred to as "ICAT", and the present invention will be referred to as "ICAT2"..." (col.4; 15-20, emphasis added; wherein "ICAT" stands for "Interactive Code Analysis Tool");
>
> "...the existing ICAT probe design as disclosed in U.S. Pat. No. 5,901,315, to Edwards, et al., is used as a starting point, with modifications as disclosed herein in order to migrate the existing ICAT to support JPDA" (col.4: 23-31, emphasis added).

Accordingly, Evans' teaching (new features in ICAT2 and existing features of Interactive Code Analysis Tool, ICAT, which actually mean the overall features of ICAT2) directs to a single method and system which:

> "... provide improved source-level debugging capabilities of an object-oriented application program which may include linked native language dynamic load libraries ..." (emphasis added, see col.3-4, Summary of the invention).

Furthermore, wherein the reference Evans is used under 35 USC 102 rejection and set forth as a single invention (i.e., "the present invention will be referred to as "ICAT2", col.4: 19-20), however, Appellant merely pointed out different locations recited in the rejection and submitted "...that such parsing of Evans is improper and does not satisfy the prima facie showing required for establishing anticipation ..." (Brief, page 9, second paragraph) without any analysis to show why and how "cited art is not 'arranged as in the claim'".

Accordingly , Appellant's argument amounts to mere statements and assertions, which are not directed to any figures/text portions relied by the examiner in the rejection, nor reply to every ground of rejection, nor point out the supposed errors in said figures/text portions as set forth in the previous Office action – see MPEP, Appendix R - Patent Rules, Action by Applicant and Further Consideration, 37 C.F.R. section 1.111 (b).

B) Claimed Features are not Met by the Cited Art (Brief, pp. 10-12, claims 1-4):

As an initial matter, the examiner notes that Appellant explicitly set forth a method for testing an application that includes emulation language code Java and native language code C/C++, as reproduced below from originally filed specification with emphasis added:

> page 2, lines 22-23: "While Java applications can provide significant advantages, there is often advantages to utilizing native language code to perform some operations";
>
> page 7, lines 18-23: "Emulation and native language interface testing method 10 simulates interfacing problems in a test

> mode and examines the response to the simulated interfacing problems. For example, potential problems that can be encountered when enabling interaction between emulation language code (e.g., Java) and native language code (e.g., C, C++, etc.) are simulated"; and
>
> page 9, lines 17-19: "In step 120, a Java application with simulated JNI problems is run if the JNI test mode is enabled. Running the application with simulated JNI problems facilitates detection of potential issues with the JNI code".

The examiner also notes that in the specification, there is no definition for the term "wrapping". Accordingly, the term "wrapping" is interpreted at least as --containing-- as such as wrapping / containing and/or presenting.

Per the plain language of the claim, the limitation at issue "wrapping native language code in a simulation test macro which creates simulated interfacing problems" is interpreted as:

wrapping/containing/presenting native language code (C/C++) in a simulation test macro which could create simulated interfacing problems;

wherein the application under test ("an emulation language application" as recited in claim 1, lines 5-6) includes both native language code (C/C++) and emulation language code (Java) (not a regular Java application, emphasis added).

i) The limitation "wrapping native language code in a simulation test macro which creates simulated interfacing problems" (Brief, pp. 10-11, claims 1-4):

As an initial matter, Evans set forth:

> col.1: 7-11: "The present invention relates generally to computer software development tools and more particularly to a method and system for debugging a Java application that includes native method dynamic load libraries (e.g., C or C++

code)" (testing/debugging problems including interfacing problems in an application that includes both emulation language code Java and native language code C/C++).

Evans explicitly teaches:

"*wrapping native language code in a simulation test macro*" (e.g.,

col.1: 7-11: the application under test includes both emulation language code Java and native language code C/C++ (emphasis added);

FIG. 4, "a simulation test macro" as the probe function 46, col.5: 65-66 (also called the launch method 46 in col.6: 3-4);

the "main" argument/parameter of the probe function/launch method 46 is set to the name of said application under test, col.5: 64 – col.6: 5 (i.e., wrapping/containing/presenting native language code C/C++ and emulation language code Java included in said application under test in the probe function/launch method 46, emphasis added);

said simulation test macro (the probe function/launch method 46) creates simulated interfacing problems (e.g.,

"The probe was preferably implemented as two distinct processes, a first "daemon" process that performed native method debugging, and a second "probe" process that performed the Java method debugging. The first process also preferably controlled the second process across the system debug API and communicated therewith via a socket connection to facilitate the simultaneous debugging of the Java and C/C++ code comprising the target application" (col.2: 15-22, testing/debugging interfacing problems in a mixed language application that comprises both emulation code Java and native language code C/C++, i.e., without facilitating means – interfacing problems - emphasis added);

"A method and system are disclosed which provide improved source-level debugging capabilities of an object-oriented application program which may include linked native language dynamic load libraries. The improved debugger is compatible with the Java Platform Debugger Architecture ("JPDA"), and provides new capabilities such as patching of Java variables and reading and writing strings from and to the application under test and being run by a local or remote Java Virtual Machine" (col.3: 66 – col.4: 7, emphasis added);

"In as much as the existing ICAT does not support JPDA, and whereas JPDA provides improved functionality and stability for Java application program development, testing, and debugging, there is a need in the art for a system and method such as ICAT which supports JPDA and its new functionality" (col.3: 17-22, emphasis added);

FIG. 6, Interface Code Analysis Tool 41 (ICAT2) with event handlers 61-67;

ICAT2 and exception handlers (col.6: 11-21; col.8: 15-36).

ii) The limitation "*examining reaction to said simulated interfacing problems when an emulation language application is run*" (Brief, pp. 10-11, claims 1-4):

The examiner respectfully disagrees with Appellant's assertions. Evans explicitly teaches "*examining reaction to said simulated interfacing problems when an emulation language application is run*" (e.g.,

FIG. 1, JVM 16 running application 17, col.4: 65 – col.2: 17;

"ICAT2 allows the user to choose whether or not to have the debugger report exceptions that occur within "try" blocks. When an exception occurs outside a "try" block, execution always stops and the exception is reported to the user.

... When ICAT2 calls "createExceptionRequest", one of the parameters to the method is a Boolean which indicates whether caught exceptions (exceptions within a "try" block) are to be reported or ignored. The newly created "ExceptionRequest" object is enabled after setting the suspend policy" (col.8: 15-36, emphasis added);

examining reports of said exceptions, events, bugs as interfacing problems; setting breakpoints, stepping applications, and examining application stack and variables, col.2: 33-47; col.3: 66 – col.4: 60).

2. Whether Claims 5-8 are Anticipated Under 35 USC 102(e) by Evans (Brief, pp. 13-16):

A) Cited Art is not "Arranged as in the Claim" (Brief, pp. 13-14):

The examiner respectfully disagrees with Appellant's assertions. As an initial matter, the examiner notes that these issues have been raised and fully responded in the previous Office action mailed April 30, 2007.

In the Remarks previously filed February 6, 2007, The Appellant asserted, "...the present rejection picks and chooses information from a variety of locations and embodiments/examples within the Evans reference in an attempt to assemble the elements of the invention as claimed ..." (Remarks, page 11: 24-26, page 12: 25 – page 13: 2, and page 13: 21-25).

In contrast with Appellant' assertions, Evans explicitly set forth:

> "...By modifying the currently-available ICAT to use JPDA, users of the new ICAT have access to the new features immediately and in future ICAT releases. For clarity and distinction, the existing debugger tool will be referred to as "ICAT", and the present invention will be referred to as "ICAT2"..." (col.4; 15-20, emphasis added; wherein "ICAT" stands for "Interactive Code Analysis Tool");
>
> "...the existing ICAT probe design as disclosed in U.S. Pat. No. 5,901,315, to Edwards, et al., is used as a starting point, with

> modifications as disclosed herein in order to migrate the existing ICAT to support JPDA" (col.4: 23-31, emphasis added).

Accordingly, Evans' teaching (new features in ICAT2 and existing features of Interactive Code Analysis Tool, ICAT, which actually mean the overall features of ICAT2) directs to a single method and system which

> "... provide improved source-level debugging capabilities of an object-oriented application program which may include linked native language dynamic load libraries ..." (emphasis added, see col.3-4, Summary of the invention).

Furthermore, wherein the reference Evans is used under 35 USC 102 rejection and set forth as a single invention (i.e., "the present invention will be referred to as "ICAT2", col.4: 19-20), again, Appellant merely pointed out different locations recited in the rejection and submitted "...that such parsing of Evans is improper and does not satisfy the prima facie showing required for establishing anticipation ..." (Brief, page 14, first paragraph) without any analysis to show why and how "cited art is not 'arranged as in the claim'".

Accordingly , Appellant's argument amounts to mere statements and assertions, which are not directed to any figures/text portions relied by the examiner in the rejection, nor reply to every ground of rejection, nor point out the supposed errors in said figures/text portions as set forth in the previous Office action – see MPEP, Appendix R - Patent Rules, Action by Applicant and Further Consideration, 37 C.F.R. section 1.111 (b).

B) Claimed Features are not Met by the Cited Art (Brief, pp. 14-16, claims 5-8):

The limitations "*means for storing said information, including said instructions for testing said Java Native Language Interface, said means for storing said information coupled to said means for communicating information*" (pp. 10-11):

The examiner respectfully disagrees with Appellant's assertions. Evans teaches:

"means for communicating information" (e.g.,

FIG. 6, Interface Code Analysis Tool 41 (ICAT2) with event handlers 61-67;

"ICAT2 allows the user to choose whether or not to have the debugger report exceptions that occur within "try" blocks. When an exception occurs outside a "try" block, execution always stops and the exception is reported to the user.

... When ICAT2 calls "createExceptionRequest", one of the parameters to the method is a Boolean which indicates whether caught exceptions (exceptions within a "try" block) are to be reported or ignored. The newly created "ExceptionRequest" object is enabled after setting the suspend policy" (col.8: 15-36, emphasis added);

examining reports of said exceptions, events, bugs as interfacing problems, setting breakpoints, stepping applications, and examining application stack and variables, col.2: 33-47; col.3: 66 – col.4: 60

GUI to set breakpoints, step applications, examine application stack and variables, col.2: 33-47;

i.e., reports, exceptions, events, breakpoints, stack values, variables ... as information)

"means for storing said information, including said instructions for testing said Java Native Language Interface, said means for storing said information coupled to said means for communicating information" (e.g.,

col.6: 52-57; col.8: 15-19; col.8: 25-col.9: 67; FIG. 13, data sent to application, data output from application;

two distinct processes comprising a first "daemon" process that performed native method debugging and a second process that performed Java method debugging, col.2: 15-22;

means for storing said information in a computer, col.4: 65 – col.5: 17, "ICAT2 on a Single System";

means for storing said information in host and target systems, col.5: 18-46, "Running ICAT2 on Two Systems").

3. Whether Claims 9-20 are Anticipated Under 35 USC 102(e) by Evans (Brief, pp. 17-20):

A) Cited Art is not "Arranged as in the Claim" (Brief, pp. 17-18):

The examiner respectfully disagrees with Appellant's assertions. As an initial matter, the examiner notes that these issues have been raised and fully responded in the previous Office action mailed April 30, 2007.

In the Remarks previously filed February 6, 2007, The Appellant asserted, "...the present rejection picks and chooses information from a variety of locations and embodiments/examples within the Evans reference in an attempt to assemble the elements of the invention as claimed ..." (Remarks, page 11: 24-26, page 12: 25 – page 13: 2, and page 13: 21-25).

In contrast with Appellant' assertions, Evans explicitly set forth:

> "...By modifying the currently-available ICAT to use JPDA, users of the new ICAT have access to the new features immediately and in future ICAT releases. For clarity and distinction, the existing debugger tool will be referred to as "ICAT", and the present invention will be referred to as "ICAT2"..." (col.4; 15-20, emphasis added; wherein "ICAT" stands for "Interactive Code Analysis Tool");
>
> "...the existing ICAT probe design as disclosed in U.S. Pat. No. 5,901,315, to Edwards, et al., is used as a starting point, with modifications as disclosed herein in order to migrate the existing ICAT to support JPDA" (col.4: 23-31, emphasis added).

Accordingly, Evans' teaching (new features in ICAT2 and existing features of Interactive Code Analysis Tool, ICAT, which actually mean the overall features of ICAT2) directs to a single method and system which

> "... provide improved source-level debugging capabilities of an object-oriented application program which may include linked

native language dynamic load libraries ..." (emphasis added, see col.3-4, Summary of the invention).

Furthermore, wherein the reference Evans is used under 35 USC 102 rejection and set forth as a single invention (i.e., "the present invention will be referred to as "ICAT2", col.4: 19-20), again, Appellant merely pointed out different locations recited in the rejection and submitted "...that such parsing of Evans is improper and does not satisfy the prima facie showing required for establishing anticipation ..." (Brief, page 18, second paragraph) without any analysis to show why and how "cited art is not 'arranged as in the claim'".

Accordingly , Appellant's argument amounts to mere statements and assertions, which are not directed to any figures/text portions relied by the examiner in the rejection, nor reply to every ground of rejection, nor point out the supposed errors in said figures/text portions as set forth in the previous Office action – see MPEP, Appendix R - Patent Rules, Action by Applicant and Further Consideration, 37 C.F.R. section 1.111 (b).

B) : Claimed Features are not Met by the Cited Art (Brief, pp. 18-20, claims 9-20):

The limitations "*running a Java application with simulated Java Native Language Interface problems if said Java Native Language Interface test mode is enabled*".

The examiner respectfully disagrees with Appellant's assertions. In the specification, the Appellant clearly set forth:

> "Emulation and native language interface testing method 10 simulates interfacing problems in a test mode and examines the response to the simulated interfacing problems. For example, potential problems that can be encountered when enabling interaction between emulation language code (e.g., Java) and native language code (e.g., C, C++, etc.) are simulated. In one embodiment of the present invention, emulation and native

> language interface testing method 10 is implemented on an emulation language virtual machine" (emphasis added).

In light of the specification (i.e., running test mode to simulate interfacing problems of an application comprising both emulation language code such as Java and native language code such as C, C++), Evans explicitly teaches:

"investigating Java Native Language Interface test mode status" (e.g., FIG. 4, col.5: 62 – col.6: 5, if debugging is enabled, calling "launch_then_attach" 46, passing arguments to each connector, setting the "main" argument to the name of the application ...);

"running a Java application with simulated Java Native Language Interface problems if said Java Native Language Interface test mode is enabled" (e.g.,

> FIG. 1, JVM 16 running application 17, col.5: 7-12;
>
> "The present invention relates generally to computer software development tools and more particularly to a method and system for debugging a Java application that includes native method dynamic load libraries (e.g., C or C++ code)" (col.1: 7-11, emphasis added); and
>
> "The probe was preferably implemented as two distinct processes, a first "daemon" process that performed native method debugging, and a second "probe" process that performed the Java method debugging" (col.2: 15-18, i.e., native language code is wrapped/processed by a distinct process as a first "daemon" process, emphasis added).

Accordingly, Appellant's arguments are not persuasive. Evans explicitly and fully teaches all claimed limitations and the examiner respectfully maintains ground of 102 rejection over claims 1-20.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejection should be sustained.

Respectfully submitted,

/Thuy Dao/

TUAN DAM
SUPERVISORY PATENT EXAMINER

Conferees:

Tuan Q. Dam, SPE AU2192

Eddie C. Lee, SPE 2100

EDDIE C. LEE
SUPERVISORY PATENT EXAMINER